import pandas as pd

import numpy as np

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import CountVectorizer

from tensorflow.keras.models import Model

from tensorflow.keras.layers import Input, Embedding, Dot, Flatten, Dense

# Load datasets

movies = pd.read\_csv('movies.csv') # Movie metadata

ratings = pd.read\_csv('ratings.csv') # User ratings

# Content-Based Filtering

def create\_content\_based\_model(movies):

movies['combined\_features'] = movies['genres'] + ' ' + movies['keywords']

count\_vectorizer = CountVectorizer(stop\_words='english')

count\_matrix = count\_vectorizer.fit\_transform(movies['combined\_features'])

cosine\_sim = cosine\_similarity(count\_matrix)

return cosine\_sim

# Collaborative Filtering

def create\_collaborative\_model(num\_users, num\_movies):

user\_input = Input(shape=(1,))

movie\_input = Input(shape=(1,))

user\_embedding = Embedding(num\_users, 50)(user\_input)

movie\_embedding = Embedding(num\_movies, 50)(movie\_input)

dot\_product = Dot(axes=2)([user\_embedding, movie\_embedding])

output = Flatten()(dot\_product)

model = Model([user\_input, movie\_input], output)

model.compile(optimizer='adam', loss='mse')

return model

# Matchmaking System

def recommend\_movies(user\_id, collaborative\_model, cosine\_sim, movies, ratings, top\_n=10):

user\_rated\_movies = ratings[ratings['userId'] == user\_id]['movieId']

user\_rated\_indices = movies[movies['movieId'].isin(user\_rated\_movies)].index

sim\_scores = cosine\_sim[user\_rated\_indices].mean(axis=0)

recommendations = np.argsort(sim\_scores)[::-1][:top\_n]

return movies.iloc[recommendations]

# Main Execution

def main():

# Prepare Content-Based Model

cosine\_sim = create\_content\_based\_model(movies)

# Prepare Collaborative Model

num\_users = ratings['userId'].nunique()

num\_movies = ratings['movieId'].nunique()

collaborative\_model = create\_collaborative\_model(num\_users, num\_movies)

# Train Collaborative Model

user\_ids = ratings['userId'].values

movie\_ids = ratings['movieId'].values

scores = ratings['rating'].values

collaborative\_model.fit([user\_ids, movie\_ids], scores, epochs=5, batch\_size=64)

# Recommend Movies

user\_id = 1 # Example user ID

recommendations = recommend\_movies(user\_id, collaborative\_model, cosine\_sim, movies, ratings)

print("Recommended Movies:", recommendations)

if \_\_name\_\_ == "\_\_main\_\_":

main()